Scripting and Configuration of MOOS Applications Using Lua

Ian Katz

MIT Laboratory for Autonomous Marine Sensing Systems
Presented at MOOS-DAWG July 19th, 2011
Since 2007
First up: A brief preface
Happy Anniversary!

2001-2011
10 years of MOOS
10 years later: Is the community still growing?
Student Autonomous Underwater Challenge – Europe

July 4th-10th 2011
SAUC-E Software

- OceanShell
- 3 × ROS
- CubeOS/RobLib
- From-scratch C++
- From-scratch C++/java/python
- ...

SAUC-E Software

- OceanShell
- 3 × ROS
- CubeOS/RobLib
- From-scratch C++
- From-scratch C++/java/python
- ...

...
Today’s Topic:
How I Made MOOS Work For Me
Today’s Topic:

How We Can Make MOOS Work Better For Everyone
Goals:

I. Make MOOS more user-friendly

II. Make better use of developers’ time
Making It Happen

Part 1: Scripting

Part 2: Configuration
Part 1 of 2

Scripting MOOS
(i.e. Rapid Prototyping)
This Isn’t For You If …

• You’ve never written new code on a boat
• You’ve never re-written old code on a boat
Lines of C++ in MOOS on Odyssey IV

(Not Counting Makefile Stuff)
Make It Simpler: Rapid Development

• Cut the code lines you write in half
• Stop writing CMake build files
• No cross-compiling for embedded systems
• Spend less time writing the small utilities
Make It Simpler

Spend more time doing actual research.
Lua is a fast scripting language
Lines of Code to Implement pSystemHealth

Lua

C++
How It Works

C++ Environment

- MOOSTrace()
- GetConfiguration()
- Notify()
- Iterate()
How It Works

C++ Environment

- MOOSTrace()
- GetConfiguration()
- Notify()
- Iterate()

Lua Environment
How It Works

C++ Environment

- MOOSTrace()
- GetConfiguration()
- Notify()
- Iterate()

Lua Environment

- MOOSTrace()
- GetConfiguration()
- Notify()
How It Works

C++ Environment

- MOOSTrace()
- GetConfiguration()
- Notify()
- Iterate()

Lua Environment

- MOOSTrace()
- GetConfiguration()
- Notify()

Lua File

- Tick()
How It Works

C++ Environment
- MOOSTrace()
- GetConfiguration()
- Notify()
- Iterate()

Lua Environment
- MOOSTrace()
- GetConfiguration()
- Notify()

Lua File
- Tick()
How It Works

C++ Environment
- MOOSTrace()
- GetConfiguration()
- Notify()
- Iterate()

Lua Environment
- MOOSTrace()
- GetConfiguration()
- Notify()

Lua File
- Tick()
Example Lua Script: Reading mail

```lua
-- "vars" is a list of any MOOS messages received since
-- it can be accessed as a 2-dimensional table: ordinal_
function Tick(vars)
    -- this line sorts the messages in vars by name and others.
    HAPI_Trace("We just received ", #vars, "messages")

    -- apply Lua's syntactic sugar to the msg list
    local mail = SimplifyMail(vars)

    -- if mail.DB_TIME exists, take the latest message
    if mail.DB_TIME then
        HAPI_Trace("DB Time is", mail.DB_TIME.now)
    end
```
Example Lua Script: Posting messages

```lua
--read date from shell
local fh = io.popen('date')
local datestring = fh:read("%s")
fh:close()

fh = io.popen("date +%s")
local datenumber = tonumber(fh:read("%s"))
fh:close()

--post values
HAPI_PostString("DUMMYSCRIPT_DATESTRING", datestring)
HAPI_PostDouble("DUMMYSCRIPT_DATENUMBER", datenumber)
```
This is a Win* for MOOS

*According to me
That was Part 1

Scripting MOOS
(i.e. Rapid Prototyping)
Goals:

I. Make MOOS more user-friendly

II. Make better use of developers’ time
Part 2 of 2

Configuring MOOS
It’s a lot like this.
It’s a lot like this.

ONLY WORSE
## MOOS vs Punchcard

<table>
<thead>
<tr>
<th>Feature</th>
<th>MOOS 1725</th>
<th>Punchcard 2001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invented in</td>
<td>1725</td>
<td>2001</td>
</tr>
<tr>
<td>Used for data input</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Supports quote marks</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Spaces in strings</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
Improving MOOS Config

Plug systems:
programmatically generating
MOOS configuration
Improving Punch Cards
Eliminating the Middleman
the programming language Lua
How It Works

C++ Environment

OnStartup( )

Lua Environment

Lua File

iGPS = {
  type = "garmin",
  origin_lat = 42.234,
  origin_lon = -76.333,
}

How It Works

C++ Environment

OnStartup()

Lua Environment

Lua File

iGPS = {
    type = "garmin",
    origin_lat = 42.234,
    origin_lon = -76.333,
}

How It Works

C++ Environment

OnStartup()

Lua Environment

Lua File

- Configure()
How It Works

C++ Environment

OnStartup()

Lua Environment

Lua File
- Configure()
How It Works

C++ Environment

OnStartup()

Lua Environment

Lua File

- Configure()

iGPS = {
    type = "garmin",
    origin_lat = 42.234,
    origin_lon = -76.333,
}

How It Works

C++ Environment

OnStartup()

Lua Environment

Lua File

iGPS = {
    type = "garmin",
    origin_lat = 42.234,
    origin_lon = -76.333,
}
How It Works

C++ Environment

OnStartup( )

Lua Environment

Lua File

- SomethingElegant( )
ServerHost = localhost
ServerPort = 9000

// this is a comment
ProcessConfig = pLuaCfg
{
  aString = MyStringValueNoSpaces
  aSpacedString = S P A C E S

  aDouble = 3.141

  aBoolTrue = true
  aBoolNotTrue = false

  aMultiString = a
  aMultiString = b
  aMultiString = and so on

  aNullVal =
}
ServerHost = "localhost"
ServerPort = 9000

-- this is a comment
pLuaCfg = {
    aString = "MyStringValueNoSpaces",
    aSpacedString = ("S P A C E S"):gsub(" ", ",")
    aDouble = 3.141,
    aBoolTrue = true,
    aBoolNotTrue = false,
    aMultiString = {
        "a",
        "b",
        ("and so on"):gsub(" ", ",")
    },
    aNullVal = nil,
}
Also, it’s backwards compatible with the MOOS API.
Also, it’s backwards compatible with the MOOS API.

I have a patch that demonstrates it.
That was Part 2

Configuring MOOS

(...Better than Punch Cards)
This Has Been:

How We Can Make MOOS Work Better For Everyone
Ask Me Anything